Optimisation Of Automated Prototype Of ITSM Life Cycles Using Serverless Computing And Workflow Management

Prashant Krishnan V
Department of Computer Science and Engineering
MS Ramaiah Institute of Technology
Bangalore, India
gillyprash29@gmail.com

Vishnu Deepak
Department of Computer Science and Engineering
National Institute of Technology
Trichy, India
vishnu.tdeepak@gmail.com

Stuti Pandey
Department of Computer Science and Engineering
Delhi Technological University
New Delhi, India
stuti.2996@gmail.com

Sai Kolluri
Leader, GTS Labs Solutions
IBM
Bangalore, India
Saikolluri@in.ibm.com

***Abstract*—As the IT industry moves towards serverless computing, modularisation and event driven programming takes prominence and emerges as a powerful paradigm to follow in the design of life cycle handling routines. Serverless programming is dynamically scalable and abstracts away the resource allocation logic and brings back the focus of the programmer to the task at hand. By controlling such a platform through a rich Workflow Management tool, we are able to gleam important statistical information, and have greater control and understanding over massive amounts of data that would be collected through the daily functioning of massive clusters of servers. We illustrate our findings as we build a singular use-case scenario for exploring the applicability of the programming paradigm in this field.**

***Index Terms*—Serverless, IT, Workflow, cloud,**

I. Introduction

In a real-world scenario where tens of thousands of servers require maintenance, decommissioning and upgrading, there are several safety checks and algorithms which to be followed in order to ensure that no data is lost or no system lands in an inconsistent state. Furthermore, haphazard operations may result in sudden workload spikes and may even result in entire operations being shut down. Depending upon the scale, this can lead to days, perhaps weeks of downtime and may even put lives at risk. For avoiding such scenarios, extremely stringent protocols are usually followed when it comes to the IT industry. Layer upon layer of checks are used to ensure that absolutely no issues arise outside of a predicted, tolerable amount. However, these checking mechanisms often require human intervention and are subject to bias and error. Furthermore, throughput of an organisation greatly suffers when as the upper limits on number of tasks that can be accomplished per day are dependent upon the capability, efficiency and sheer number of personnel assigned to each task. This is where the concepts of serverless computing and event-driven programming come in. The former is a platform which takes out the job of resource allocation and management from the programmers. It was designed with the idea that the programmers need only take care of their functional logic. Event-driven programming is supported by a trigger-rule-action mechanism. A trigger is any name that is associated with an event. An action consists of instructions which are carried out after the system senses that a trigger has been fired. A rule is how triggers are mapped to actions. Actions are mapped to triggers in a many-to-one fashion. The structure of IT Service management lends itself exceptionally well to the framework of serverless computing. Each stage of approval required to be completed before a task can be completed can fire a trigger, which using a rule, will invoke an action, which will fire the next trigger and so on. If multiple lines of checks for different determinants need to take place, then the many-to-one functionality flawlessly executes the requirements. Another big positive is the fact that serverless computing facilitates function isolation using containerised environments. Therefore no function can interfere with the functioning of another function which is happening simultaneously, thereby eliminating the possibility of the functions leaving the system in an inconsistent state. Serverless computing also boasts the functionality of dynamic scaling according to the workload. That is, the system will allocate just enough resources to cater to the task size and will only start allocation when it receives explicit instructions to do so. As heavier workloads come in, the environment automatically allocates more resources to solve the tasks at hand, thereby making sure that there are no slowdowns for the end user. One of the biggest drawbacks of serverless computing is that platforms which provide this functionality do not maintain any state information. That is, serverless computing is inherently stateless and no information can be derived of their workings until the final output is seen. This is where an efficient Workflow Management tool comes in, which can be used to puppeteer and record the triggers and actions which are being used. This allows us to gleam important statistical data about how the system functions and gives us meaningful insight into trends in parameters like server usage, failure rate, workload distribution etc. all from the same platform. The remainder of the paper illustrates the functioning of our use-case and talks about the specific tools and platforms which were essential in our study of applicability of the architecture.

II. Tools Used

A. Apache OpenWhisk

Apache OpenWhisk is an open source implementation of event based service. It listens for the firing of triggers and invokes the appropriate actions based on rules which link a specific trigger to an action. This format lets OpenWhisk act as a platform for event-based/event-driven programming paradigms that leverage the power of the cloud to offer dynamic scaling functionality.The burden of managing infrastructure is lifted up from the shoulders of developers and they can exclusively focus on writing scripts to execute the application logic and creating actions that are executed on demand.Hence,Openwhisk triggers rules and actions forms the cornerstone of event- driven capabilities.

B. ServiceNow

ServiceNow is a an IT Service Management platform which was designed to incorporate facilities for a seamless workflow in the service management interface. It provides an integrated ticketing mechanism and also leverages the power of the cloud and artificial intelligence to provide smart analytics. The system facilitates incident management, problem management, change and release management, configuration management and request management among others. The details of servers which are currently operational are listed in the CMDB or Configuration Management Database. The CMDB contains several attributes like model, RAM, manufacturer and provision date among others, which can be dynamically modified according to Change Requests filed by the users. The entire Servicenow platform is used to provide a clean interface where users can manage requests and pull information about the systems in place very easily.

C. SoftLayer

IBM SoftLayer is a public cloud computing platform that offers a wide spectrum of services, including those for computing, networking, storage, security and application development. Cloud administrators and users access IBM SoftLayer services over the Internet or through a dedicated network connection.The IBM Cloud by being an open initiative,renders flexibility and control needed for leveraging skills for expansion of business. IBM SoftLayer is largely considered infrastructure as a service (IaaS), a form of cloud computing in which a third-party provider hosts hardware, software and other infrastructure components on its users' behalf.IBM Cloud offers powerful and robust hardware to meet any requirement and budget.IBM Softlayer provides smooth and quick deployment of IaaS bare metal or virtual server in couple of minutes. It also allow hassle free customization of bare metal server in 2-4 hours.The quick deployment speed of virtual servers keeps business moving even when more resources are needed on the fly.With IBM IaaS, most complex, compute-intensive workloads from analytics and graphics to energy exploration and machine learning can be handled with ease.

D. PysNow

Pysnow is an open source python library through which CRUD operations can be performed on Servicenow data which is stored in the form of tables. For the purpose of the demonstration, pysnow functions are used along with proper authentication to connect to the Servicenow instance to carry out the desired actions.

E. Vault

Vault is a Hashicorp product which secures and stores key information as secrets on a server. The server remains in a sealed state until unsealing keys (generated at the time of server initialisation) are entered sequentially. Once unsealed, authentication to access the secrets stored inside is done through a token system. The root token user can set number of use times, time limits and also the level of access of each token upon time of generation. Vault can be run using another product Hashicorp Consul as the backend.

F. AirFlow

An open source Apache project developed by Airbnb. It is a platform to programmatically author, schedule and monitor workflows. It uses the concept of directed acyclic graphs (DAGs).

III. Implementation

There are broadly 5 tasks that are part of the Airflow DAG. These tasks constitute the entire workflow. The DAG is automatically triggered by the ServiceNow portal once a new request is entered.

A. Task 1 : Obtaining required request IDs

This is the first task when the DAG is triggered. This task calls an OpenWhisk action which gets all the request IDs which have the short description ”Provision vm”. The OpenWhisk action sends the result to Airflow and using the XCom operator we forward data between tasks. Thus, the result containing request IDs from the ServiceNow portal is forwarded to the next task.

B. Task 2 : Updating status of short description and provisioning VM

This task is divided into 2 subtasks. The first one calls an OpenWhisk action which updates the status of the short description in the ServiceNow Portal to ”In progress” for the respective request. The next part of the task is probably the heaviest action of the entire workflow. This subtask consists of an OpenWhisk action which contacts SoftLayer and begins the process of provisioning a VM with the desired specifications. The request ID, asset tags and the specifications of the VM are the results returned by OpenWhisk to Airflow. These details using the XCom operator are subsequently forwarded to the next task.

C. Task 3 : Updates the details of the VM in the CMDB

The various specifications of the VM are obtained from Task 2 using the XCom operator as discussed above. We use these details and update the CMDB (Configuration Management Database) on the ServiceNow portal by invoking an OpenWhisk action which performs the same. This process of updating the details to the CMDB is useful as the details of the provisioning of the VM would be available to the client through the ServiceNow portal.

D. Task 4 : Changes the status of short description

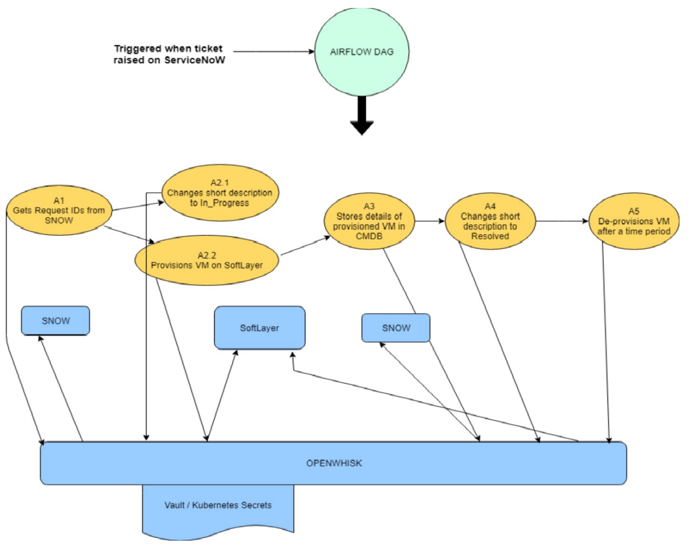
This gets the details of the request ID from the previous tasks. It calls an OpenWhisk action which updates the status of the short description to ”Resolved”. Thus, the user on the ServiceNow portal is intimated of his request being completed successfully.

E. Task 5 : De-provisions the server requested

This task is one that is a possible inclusion to the workflow based on necessary conditions. It deals with de-provisioning the VM that was earlier provisioned. There could be various conditions like based on time, say, where a client wishes to issue a VM only for a month’s time. This can be easily handled by the Airflow Scheduler and thus it will ensure that the de-provisioning happens smoothly after a month. The whole process is thus automated and the lifecycle completes. The task obtains details of the VM provisioned from Task 2. The task involves invoking an OpenWhisk action which contacts SoftLayer to de-provision the specific VM. This ends the lifecycle and once the de-provisioning is a success, Airflow will change the status of the DAG to success.

IV. Optimisation

The monitoring process of Servicenow to check for incoming requests can be eliminated by setting up a server side script which will fire based on the trigger that a new request comes in with. This is done by creating a REST message and configuring it to send HTTP POST requests to the concerned API endpoint. This message is further leveraged in a Business Rule which can be assigned to the requests table and can be made to run a Javascript function based on the required conditions.


Triggered when ticket raised on ServiceNoW
AIRFLOW DAG
A1 Gets Request IDs from SNOW
A2.1 Changes short description to In_Progress
A2.2 Provisions VM on SoftLayer
A3 Stores details of provisioned VM in CMDB
A4 Changes short description to Resolved
A5 De-provisions VM after a time period
SNOW
SoftLayer
SNOW
OPENWHISK
Vault / Kubernetes Secrets

References

[1] Antti Lahtela, Marko Jntti, Jukka Kaukola “Implementing an ITIL-Based IT Service Management Measurement System,” Fourth International Conference on Digital Society, 10-16 Feb. 2010

[2] Garrett McGrath, Paul R. Brenner , Serverless Computing: Design, Implementation, and Performance, 2017 IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW), 5-8 June 2017